Capítulo ONZE

Referências a Métodos

Objetivos do Exame:

Usar referências a métodos com Streams.

Usando métodos como objetos

Como sabemos, em Java podemos usar referências para objetos, seja criando novos objetos:

Ou usando objetos existentes:

Mas e quanto a uma referência a um método?

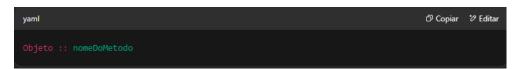
Se usarmos apenas um método de um objeto em outro método, ainda temos que passar o objeto inteiro como argumento. Não seria mais prático passar apenas o método como argumento? Algo como:



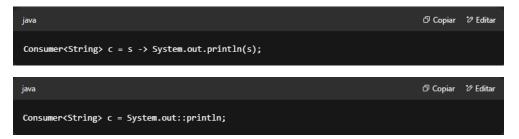
No Java 8, graças às expressões lambda, podemos fazer algo parecido. Podemos usar métodos como se fossem objetos ou valores primitivos.

E isso é possível porque uma **referência a método** é a **sintaxe abreviada para uma expressão lambda** que executa apenas **UM único método**.

Uma Referência a Método



Sabemos que podemos usar expressões lambda no lugar de classes anônimas. Mas às vezes, a expressão lambda é apenas uma chamada a algum método, por exemplo:



Numa referência a método, você coloca o **objeto (ou classe)** que contém o método antes do operador ::, e o **nome do método** depois dele, **sem argumentos**.

Mas talvez você esteja pensando:

- Como isso é mais claro?
- O que acontece com os argumentos?
- Como isso pode ser uma expressão válida?
- Não entendo como construir uma referência a método válida.

Primeiramente, uma referência a método não pode ser usada para qualquer método.

Ela pode ser usada apenas para substituir uma expressão lambda de um único método.

Portanto, para usar uma referência a método, você precisa primeiro de uma expressão lambda com um único método.

E para usar uma expressão lambda, você precisa de uma **interface funcional**, uma interface com **apenas um método abstrato**.

Em outras palavras:

Em vez de usar

UMA CLASSE ANÔNIMA

você pode usar

UMA EXPRESSÃO LAMBDA

E se esta chamar apenas um método, você pode usar

UMA REFERÊNCIA A MÉTODO

Existem quatro tipos de referências a métodos:

- 1. Uma referência a um método estático
- 2. Uma referência a um método de instância de um objeto de um tipo específico
- 3. Uma referência a um método de instância de um objeto existente
- 4. Uma referência a um construtor

Referência a um método estático

Vamos revisar um exemplo do capítulo anterior:

Saída:

```
36
30
```

A chamada Reference1::add retorna uma referência ao método add() da classe Reference1.

Esse método precisa ter a mesma assinatura que a da interface funcional BiFunction.

Referência a um método de instância de um objeto de um tipo específico

Considere o seguinte exemplo:

```
public class Reference2 {
   public boolean isGT10(int i) {
      return i > 10;
   }

   public static void main(String[] args) {
      Reference2 r = new Reference2();

      IntPredicate ip1 = i -> r.isGT10(i);
      System.out.println(ip1.test(5));

      IntPredicate ip2 = r::isGT10;
      System.out.println(ip2.test(15));
   }
}
```

Saída:

```
false true
```

Observe que aqui usamos um método não-estático (isGT10) e um objeto já existente (r) para chamá-lo.

Nesse caso, a chamada r::isGT10 retorna uma referência ao método de instância isGT10() do objeto r.

Referência a um método de instância de um objeto de um tipo arbitrário

Esse é um pouco mais complicado de entender. Veja:

```
public class Reference3 {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("a", "b", "A", "B");

        list.sort((s1, s2) -> s1.compareToIgnoreCase(s2));
        System.out.println(list);

        list.sort(String::compareToIgnoreCase);
        System.out.println(list);
    }
}
```

Saída:

Aqui usamos o método compareTolgnoreCase, que é um método de instância da classe String.

Mas não o invocamos com um objeto específico. Em vez disso, dizemos que vamos usá-lo **em qualquer objeto String**, então a sintaxe é:



A diferença aqui é que o **primeiro argumento da lambda** (s1) será o objeto que chama o método, e o **segundo argumento** (s2) será passado para o método.

Ou seja, a lambda:

é equivalente a:



Referência a um construtor

Considere o seguinte exemplo:

Saída:

No primeiro caso, usamos uma expressão lambda para invocar o construtor.

No segundo caso, usamos uma **referência a construtor** com Reference4::new, que é equivalente à lambda () -> new Reference4().

Pontos-chave

- Uma referência a método é uma forma abreviada de uma expressão lambda que invoca um único método.
- A sintaxe para uma referência a método é:

- As referências a métodos só podem ser usadas se houver uma **interface funcional compatível** com a assinatura do método referenciado.
- Há quatro tipos de referências a métodos:
 - 1. Referência a método estático:

ClassName::staticMethodName

2. Referência a método de instância de um objeto existente:

object::instanceMethodName

3. Referência a método de instância de um tipo arbitrário:

ClassName::instanceMethodName

Neste caso, o primeiro argumento da lambda se torna o receptor (this) da chamada de método.

4. Referência a construtor:

ClassName::new

Autoavaliação (Self Test)

1. Dado o seguinte código:

```
public class Question_11_1 {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("1", "2", "3");

        list.forEach(system.out.print(s + " "));
        list.forEach(System.out::print);
    }
}
```

Qual é a saída?

A. 123

B. 123

C. 1 2 3123

D. 123123

E. A compilação falha

2. Qual das seguintes afirmações é verdadeira?

- A. Uma referência a método só pode ser usada se a interface funcional contiver um método apply()
- B. Uma referência a método pode ser usada para qualquer método, desde que os nomes combinem
- C. Uma referência a método é uma forma abreviada de uma expressão lambda que invoca um único método
- D. Uma referência a método pode ser usada para qualquer interface com métodos default

3. Qual é a saída do seguinte código?

```
public class Question_11_3 {
    static boolean isEven(int i) {
        return i % 2 == 0;
    }

    public static void main(String[] args) {
        IntPredicate ip = Question_11_3::isEven;
        System.out.print(ip.test(10));
    }
}
```

- A. true
- B. false
- C. 0
- D. 1
- E. A compilação falha

4. Qual é a saída do seguinte código?

```
java

public class Question_11_4 {
    public static void mair(String[] args) {
        BiPredicate<String, String> bp = String::equalsIgnoreCase;
        System.out.print(bp.test("java", "JAVA"));
    }
}
```

- A. true
- B. false
- C. JAVA
- D. java
- E. A compilação falha

5. Qual é o nome técnico da seguinte forma de referência?

- A. Referência a método estático
- B. Referência a método de instância de objeto arbitrário
- C. Referência a método de instância de objeto específico
- D. Referência a construtor